Love and Marriage: CMMI and Agile Need Each Other

Hillel Glazer
Entinex, Inc.

Agile's values and practices ensure critical, long-term process success, making it an ideal partner of the CMMI[1] framework, which delivers a robust infrastructure of organization-wide, broadly inculcated continuous improvement and optimization. Intended for an audience of process improvement professionals, CMM left out some of the basic elements critical to long-term process success [1] *that—as luck would have it—Agile values and practices supply. Together, Agile and CMMI complete each others' capabilities and can lead to fast, affordable, visible, and long-term benefits.*

In early 2001, the Agile movement can be arguably said to have gelled with the formulation of the "Manifesto for Agile Software Development" (often called the Agile Manifesto) [2]. Other than two items published later that year—one in CROSSTALK [3] and another in *IEEE Software* [4]—much of the writing from 2001-2008 on the topic of CMM [5, 6] (or CMMI) and Agile development had been limited to online sources such as e-mail groups, user forums, blogs, and wikis[2]. In that time, much of what was written on the topic was mostly on how the two bodies of ideas were incompatible.

Barry Boehm and Richard Turner's "Balancing Agility and Discipline" [7] provides a sound, practical, robust risk-based approach to reconciling what was widely perceived as being orthogonal interests of discipline, à la CMMI (then v1.1) and *agility* (of any variety). Nonetheless, the *process myth* of irreconcilability between CMMI and Agile persisted. User-group-type organizations—such as the SEI's Systems and Software Process Improvement Network and Agile Project Leadership Network (APLN) groups—were fostering gatherings based on the topic, but many were billed as *confrontational* panel discussions and/or *contrarian* viewpoints [8]. Over time, however, the topic began to become more seriously inspected, fueled by curiosity as described in [9, 10, 11].

There have been various missteps made throughout the existence of CMMI and its predecessors [12] that contributed to the creation of the Agile Manifesto [13]. Unfortunately, these missteps will continue to be made by people who are not appropriately qualified to be using, appraising, or teaching CMMI (more on that later). Nonetheless, making progress—in a world that accepts structured, deliberate, and persistent improvements together with empowered teams, Lean processes, experimentation, and involved customers—requires that all parties:

- Understand the foundation and intent of both CMMI and Agile.
- Implement CMMI and Agile goals, values, and practices in synergistic ways.

Both Agile and CMMI have been shown to benefit project and organizational performance. To gain the maximum results of a combined approach, appropriate expectations from both must be set. Before this can be done, we must also incorporate particular context of CMMI and Agile.

> *"CMMI focuses on practices and artifacts of cultures of process excellence without addressing the underlying enablers of this culture."*

CMMI Isn't for Everyone—It's for Experts

Whether intentional or not, the body of work that is the CMMI and accompanying products and services (appraisals, training, etc.) in large part targets to an audience of subject-matter experts (SMEs): people with knowledge, training, skills, and a foundation of process improvement theory and practice. There is a tacit assumption that they have broad, practical, applied improvement experience in their particular domain of work and are *professionals* in process improvement. For process improvement SMEs, CMMI is abstract enough, but for people without *a priori* background, experience, and education in process improvement, CMMI is hard to use and lacks sufficient context and background on fundamental process improvement. CMMI, perhaps rightly so, doesn't regress to explain the foundation from which it emerged [1].

CMMI focuses on practices and artifacts of cultures of process excellence without addressing the underlying enablers of this culture. In fact, the term *culture* in any form appears only sparingly in CMMI: twice in reference to choosing a *model representation* (which only really matters when pursuing appraisals, not improvements); once buried in an example within a subpractice of the Organizational Innovation and Deployment process area (which is at Maturity Level 5); once in an example in the introductory notes to the Process and Product Quality Assurance process area; and once in the glossary definition of *institutionalization*. As well, there is no discussion of how to attain a culture conducive to process improvement or what its attributes are.

Although built on decades of process improvement practice—from Deming, Juran, and Crosby to Ohno, Shingo, and the work at Toyota[3]—CMMI doesn't mention these thought-leaders, nor does CMMI directly explain that their work in creating and fostering high-performance organizations is both context to and reflected in CMMI. While not explicitly prerequisites to using CMMI, it stands to reason that having the basic knowledge, technique, and application of process improvement makes success with using CMMI more achievable. The audience for CMMI is assumed to have a working knowledge of and background in basic process improvement.

Just as there are no actual processes in CMMI—and none of the process

areas have enough content to actually create fully functioning processes for actually developing real products and services—CMMI similarly doesn't include the principles and practices of basic process engineering and design. Again, this entry-level content isn't something one would expect to see in an advanced work on process improvement; nonetheless, when people start with CMMI without this context, their failure is predictable.

The contents of the People Capability Maturity Model [14] aren't part of CMMI either. Organizations believe they can achieve high value in CMMI implementations by focusing exclusively on CMMI while ignoring workforce competency and empowerment facets of a process improvement culture (which happen to be among the Agile principles). The attributes and capabilities of high-performing, strongly process-oriented, and well-integrated cultures of improvement are captured in [15].

Agile Limitations (That CMMI Can Mitigate)

Agile values, principles, and practices have been demonstrably beneficial to many organizations. The story is not all rosy, however, and there are plenty of examples of Agile failing to achieve desired outcomes just as there are failures of CMMI to achieve desired outcomes[4]. Failures with Agile often have similar causes to those in CMMI: failing to account for context, background, and culture, and implementing incomplete components.

However, some Agile failures can be mitigated with the practices and constructs in CMMI. Agile values, principles, and practices are mostly oriented at the team and project level and their premises rely heavily on individuals and those currently and immediately involved in a particular effort. CMMI goals and practices assume an organization wants its processes propagated widely and over distances of time and/or space. Therefore, CMMI provides an organizational-level infrastructure as well as mechanisms to:

- Preserve information and knowledge over time.
- Structure and provide criteria for decision-making.
- Strengthen and normalize risk management.
- Methodically apply technical approaches.
- Specifically focus on process improvements.
- Specify engineering practices. (CMMI includes several engineering best practices, but assumes users know the basics).

Additionally, CMMI includes practices for the normalization of processes at the organizational and project levels that are, upon closer inspection, attributes of an organizational culture of process performance. These *generic practices* are frequently overlooked by organizations using Agile approaches since the focus of Agile values is too often applied in a short-term and target effort view, frequently failing to take into account the value of the *things on the right* [2, 13].

> *"Failures with Agile often have similar causes to those in CMMI: failing to account for context, background, and culture, and implementing incomplete components."*

Another area where Agile's content clearly has no material is in the area of quantitative process performance. Some might argue that there is no point to pursuing quantitative optimization since the very notion assumes a process is performed with enough frequency and regularity that pursuit of a quantitative model is value-added.

This is a valid argument, especially for: projects that never have anything in common with other projects; projects that are short-term and/or require minimal effort; organizations that band and disband in an ad-hoc fashion as a function of project scope and client need; and organizations that don't understand that quantitative performance need not be onerous [16]. However, for projects involving more than one or two people, that take months, that have teams whose resource pool (of a dozen or so) is stable and reusable, and for organizations that approach development valuing certain aspects of performance predictability, process optimization is neither out-of-reach nor pointless. In fact, processes can be described, stabilized, normalized, capable, baselined, and optimized with just a few iterations or sprints.

With so much of Agile's content drawing from Lean sources, it is conspicuously lacking in quantitative techniques that are so much of a staple in those very same practices. Concepts such as TQM, Six Sigma, and the Toyota Production System [17, 18, 19]—from which several highly valued Agile principles are based—are all more than principle and culture alone: They are deeply quantitative and steeped in detailed process definitions and standards. Yes, the culture preceded the process definition and statistics, but the process definition and statistics are also a reflection of the culture as well as a facilitator of the relentless pursuit of customer delight. Many practitioners of Agile values and principles stop well short of quantitative techniques and process definitions, thereby making an unwitting shortfall in their own pursuits of excellence.

This is another role played by misinterpretation of CMMI. *Processes* (in the large) are not quantitatively characterized, but rather the focus is on *subprocesses* [20]. Measures are taken at specific points, not at all process junctions, and not throughout a process' use, but where it makes business sense and adds organizational value. With this in mind, even Agile organizations will find that there are many activities they perform with regularity even when projects, teams, and customers are different. Some examples are:

- Refactoring.
- Continuous integration.
- Test-driven development.
- Sprint/iteration planning.
- Planning Poker estimation.
- Pair programming.

Each of these practices are not total processes; they are made-up of subprocesses which can be measured for duration, defects, effort, instances (counts), and so forth. For certain applications within certain projects, there may be benefits to normalizing and then quantifying how some of the underlying subprocesses in these practices perform. Dismissing quantification out-of-hand is nowhere to be found in the Agile values or practices.

Agile Teachings (That CMMI Can Benefit From)

Simplicity. Putting the *basics* back into the improvements.

Among the several catalysts for the

Agile movement were troubling fads in the software development world regarding processes, tools, and methodologies. In particular, these fads were fast approaching the status of trends, pulling the development world into a *vicious cycle* of under-performing project results—despite the ever-increasing presence of concepts ostensibly created to bring about success.

The emergence of the Agile movement did the software development world (and the world of process improvement, in general) a great service by reminding us of the fact that individuals and interactions, working products, customer collaboration, and responding to change do matter more than processes and tools, unnecessary documentation, contract negotiation, and continuing to follow an obsolete plan [16]. Specifically, the things that matter most to the customer are the things the customer perceives, experiences, and pays for. In other words, the things that matter to an organization are facilitated by delivering on things that matter to the customer. Therefore, customer needs are (and always have been) a higher priority to the viability of any business transaction than the *unseen machinations* that enable the organization to deliver against those customer needs.

Is this to say that the unseen machinations are entirely unimportant? No. For the long-term viability of the business, the underlying processes that make the business operate are critical. However, these are less important to the customer than is meeting their immediate expectations. A simple way to reiterate [2] in a process improvement-oriented way is to say *all efforts must align with the needs of the business to satisfy the customer*.

Another practice found among *Agilistas* is that of experimentation: a time-honored process improvement technique long ago lost among the level-manic set. Not just experimenting with solutions in the *fail early and often* Agile sense, but experimenting with processes, organizations, data collection, and tracking techniques. It's not uncommon to hear "... we don't want to try that because it might kill our level."

Mired in the unnecessary bureaucracies of formal process groups, the idea of experimentation with processes has been nearly erased from the process improvement tool kit. In CMMI, the term *experiment* appears exactly three times, not one of which is connected to basic process development. The closest reference to the term is found in the Causal Analysis and Resolution process area (which most organizations don't look at until pursuing Maturity Level 5) as a suggestion within a subpractice related to implementing "action proposals" [21].

Again, this is not a dig at CMMI—it is pointing out the extent of context and knowledge assumed to exist among CMMI users. If any criticism of CMMI in this regard is warranted, it is that the model front-matter does not highlight these assumptions. It leaves hapless project members without a map of steps and tools to navigate by, drowning in the gravity well of process improvement intricacies. The front-matter states:

> The audience for this model includes anyone interested in process improvement in a development and maintenance environment. Whether you are familiar with the concept of capability maturity models or whether you are seeking information to get started on your improvement efforts, this document will be useful to you.
>
> This model is also intended for people who want to use an appraisal to see where they are, those who already know what they want to improve, and those who are just getting started and want to develop a general understanding of the CMMI for Development constellation. [22]

"A danger posed to practitioners of both CMMI and Agile is the incorrect assumption that using either is a substitute or an excuse for ignoring the discipline inherent in actual engineering activities ..."

Without any mention of assumed capabilities, experience, training, aptitudes, knowledge, or skills, CMMI is foisted by level-hungry management on unsuspecting users. Instead, readers most likely expected sufficient content in the model explaining not only what needs to be done, but guiding them in even the most macro-level best practices during implementation. Psychological change management skills and laying the groundwork for the right culture are critical to implementing CMMI, yet are absent from the text. Organizations left without the appropriate understanding are presumed to use CMMI not knowing they lack the wherewithal to get anywhere.

But thanks to Agile's values, principles, and practices and their simple portrayal of the basics of process design and use, organizations can adopt Agile ideas—and with them implement CMMI—without first becoming masters of process improvement. Agile can prevent the bloating of processes, ensure that processes add value, involve the necessary stakeholders, encourage experimentation, and replace level mania with results.

The Perfect Marriage?

Neither CMMI nor Agile include content that replaces thorough engineering practices. A danger posed to practitioners of both CMMI and Agile is the incorrect assumption that using either is a substitute or an excuse for ignoring the discipline inherent in actual engineering activities (or appropriate activities for acquisition and/or services).

For example, an organization that needs to be taught how to analyze requirements to ensure they are necessary and sufficient, or that believes a design can be fully illuminated from conversation alone (and doesn't need description or revisiting), isn't doing engineering and is probably not ready for CMMI. Furthermore, regardless of whether or not such an organization is using CMMI or whether or not they are implementing Agile practices, it isn't actually *developing* in the engineering sense. Such organizations may be programming or coding, but they are not developing. Not performing engineering is nothing more and nothing less than not performing engineering. Having said that, not every project actually requires engineering to be done at the project level. Some projects are merely carrying out the final touches on works that have already been engineered long ago and/or by another organization, leaving the more mechanical work to others. Sadly, it is also too often that engineer-

ing is ignored when it very much ought to be done. This may point to other, deeper issues with software development in general, as raised in [23, 24, 25, 26].

Simply put, when actual engineering is being performed, it produces the appropriate artifacts. Neither CMMI nor Agile can supply enough content to cause (or make necessary) actual engineering practices.

The following examples describe a few ways in which CMMI and Agile were able to work synergistically during development activities.

In one case, Agile practices (Scrum) were already in place and operating effectively. A customer of the organization required a CMMI Maturity Level 2 rating to be compliant with their contract. Although the manager of software development believed there would be a benefit to implementing the practices all the way through Maturity Level 3, executive management preferred the lower expense and faster results of pursuing a Maturity Level 2 rating. Nonetheless, in roughly nine months, the software team of approximately a dozen cross-trained developers and a few specialists had satisfied the Maturity Level 2 goals, also (without explicitly trying) satisfying the goals of at least two or three additional process areas. Their approach included:

- Using Scrum to manage the process deployment and improvement effort and operating the process engineering group as a Scrum team.
- Using measures and metrics that were both easy to obtain and relevant indicators of process, project, organizational, and product performance.
- Integrating CMMI practices into their workflow, including expanding Scrum practices to account for all process activities (regardless of whether they were tied to CMMI or not).
- Leveraging Scrum's product and sprint backlogs, daily stand-up meetings, and end-of-iteration retrospectives for conducting activities that improve their processes.
- Creating a simple developer handbook that explains how product development took place and points its users to all the process assets necessary to do their jobs.
- Creating templates and checklists for the routine aspects of product development and project governance.
- Adding discipline and standards to areas of work that had been allowed to be performed *freestyle*.
- Rotating personnel on and off of the process group to broaden the experience base and keep interest fresh.

None of these attributes of the client's approach are necessarily unique to Scrum/Agile or CMMI; however, the key factor in the client's easy success was that they began with and applied Agile values and practices to implement CMMI, and learned through CMMI the benefits of having certain practices of theirs be more under their control.

This resulted in all projects using a single set of broad practices and measures, increasing the predictability of project activities and the ability to demonstrate progress in process effectiveness. The teams also worked more steadily with fewer disruptions by building specific *touch points* into their workflow. The team also learned that process artifacts were less valuable and more disruptive when not fully designed. Though the discipline that some process activities afforded was beneficial, the team learned that taking the *easy way out* actually proved to be more disruptive than had they created more appropriate tools.

In another case, an organization had been using many practices from Extreme Programming for some time and had experienced dramatic improvements from their previous approach. As a result, they found themselves in the most well-suited position to be leveraged for pursuing Maturity Levels 4 and 5. Rather than burdening themselves with traditional measurement techniques and objectives, the development leader was able to identify several natural *measurement points* throughout their workflow. Such natural measurement points fell into a few general areas:

- Between physical steps in the workflow.
- Wherever automation is able to collect data.
- Whenever data is being entered into or manipulated in a tool.
- During (or subsequent to) refactoring activities.
- At the beginning of iterations where analysis, estimates, and task allocation is performed.
- At the end of iterations (and at releases) where much of the progress-to-date is being reviewed.

One interesting decision was to not attempt certain metrics from the activi-

ties of pair programming. This decision was reached when the impact of the measurement effort was found to exceed the value of the measures. The ability to describe the processes affecting the measures was also determined to be too complex and therefore onerous to control. Despite enormous pressure (and opportunity) to gain valuable data to facilitate high-maturity behaviors, a business decision was made regarding which processes would add value by being optimized (after determining that it would not add business value to attempt to optimize pair programming).

Although this organization has yet to attain the necessary performance and depth of measures to approach Maturity Levels 4 and 5, the lessons they learned and applied from both Agile and CMMI are moving them towards becoming a more highly performing Agile team. One lesson that has been implemented was the realization that greater discipline and finer granularity in backlog management leads to more accurate estimates of task effort. Another lesson was that the intent of CMMI practices may already be accomplished by existing activities when such activities are viewed in broader engineering terms in addition to their more common Agile terms.

Conclusion

CMMI can't be everything to all users. Some users will work with CMMI from the perspective of already being process improvement experts and some will be novices. Regardless, it is easy to take wrong turns with CMMI. However CMMI, in the right hands, can facilitate an evolutionary path towards optimization.

Agile helps improve many operational and transactional activities but wasn't intended to provide higher levels of organizational constructs to facilitate long-term process evolution. Nonetheless, Agile can jump-start effective process design and deployment, and foster a culture of process excellence through its core values in Lean and cooperative processes.

Together, CMMI and Agile can operate synergistically to enhance the other's performance, speed to deployment, and acculturation. Organizations would be well-advised to set aside their prior perceptions of CMMI and Agile compatibility and embrace both their mutually beneficial and shared vision: delivering a high-quality product to the customer on time.◆

References

1. Konrad, Michael D. Personal interview. 2009.
2. Beck, Kent, et al. "Manifesto for Agile Software Development." Feb. 2001 <www.agilemanifesto.org>.
3. Glazer, Hillel. "Dispelling the Process Myth." CROSSTALK Nov. 2001.
4. Paulk, Mark C. "Extreme Programming From a CMM Perspective." *IEEE Software* 18.6 (Nov./Dec. 2001): 19-26 <ftp.sei.cmu.edu/pub/documents/articles/pdf/xp-from-a-cmm-perspective.pdf>.
5. Paulk, Mark C., et al. *Capability Maturity Model for Software*. Version 1.1. SEI, Carnegie Mellon University. Technical Report CMU/SEI-93-TR-024 ESC-TR-93-177. Feb. 1993 <ftp.sei.cmu.edu/pub/documents/93.reports/pdf/tr24.93.pdf>.
6. Paulk, Mark C., et al. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Boston: Addison-Wesley Professional, 1995.
7. Boehm, Barry, and Richard Turner. *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston: Addison-Wesley Professional, 2003.
8. Dalton, Jeffrey. Personal interview. 2007.
9. Glazer, Hillel. "Keys to Making Agile and CMMI Compatible" APLN – Maryland, Chapter Meeting. Audio Podcast and Slides. 15 May 2007 <http://apln.agilemaryland.org/moin/Meeting2007May>.
10. Cook, Linda M. Personal interview. 2007.
11. Dinwiddie, George. Personal interview. 2007.
12. Glazer, Hillel, et al. *CMMI or Agile: Why Not Embrace Both!* SEI, Carnegie Mellon University. Technical Note CMU/SEI-2008-TN-003. Nov. 2008 <www.sei.cmu.edu/reports/08tn003.pdf>.
13. Cockburn, Alistair. Personal interview. 2009.
14. Curtis, Bill, Bill Hefley, and Sally Miller. *People Capability Maturity Model, Version 2.0, Second Edition*. SEI, Carnegie Mellon University. Technical Report CMU/SEI-2009-TR-003 ESC-TR-2009-003. July 2009 <www.sei.cmu.edu/reports/09tr003.pdf>.
15. Spear, Steven J. *Chasing the Rabbit: How Market Leaders Outdistance the Competition and How Great Companies Can Catch Up and Win*. McGraw Hill, 2008.
16. Anderson, David. *Measurement and Analysis in Agile Methods*. Proc. of the 21st Annual SEPG North America Conference. San Jose, CA. 23-26 Mar. 2009.
17. Ohno, Taiichi. *Toyota Production System: Beyond Large-Scale Production*. New York: Productivity Press, 1995.
18. Shingo, Shigeo. *A Study of the Toyota Production System from an Industrial Engineering Viewpoint (Produce What Is Needed, When It's Needed)*. Trans. by Dillon, Andrew P. New York: Productivity Press, 1989.
19. Spear, Steven, and Bowen, H. Kent. "Decoding the DNA of the Toyota Production System." *Harvard Business Review*. Sept.-Oct. 1999 <http://twi-institute.com/pdfs/article_DecodingToyotaProductionSystem.pdf>.
20. CMMI Product Team. *CMMI for Development, Vers. 1.2*. SEI, Carnegie Mellon University. Technical Report CMU/SEI-2006-TR-008. "Organizational Process Performance" section: 261-274. Aug. 2006 <www.sei.cmu.edu/reports/06tr008.pdf>.
21. CMMI Product Team. *CMMI for Development, Vers. 1.2*. "Causal Analysis and Resolution – A Support Process Area at Maturity Level 5" section: 104-106.
22. CMMI Product Team. *CMMI for Development, Version 1.2*. "Preface – Audience" section: iii.
23. Adams, William. With response by Grady Booch. "The Relevance of Architecture." *IEEE Software* 24.5 (Sept./Oct. 2007): 8-9 <http://www2.computer.org/portal/web/csdl/abs/html/mags/so/2007/05/s5008.htm>.
24. Alleman, Glen B. "Bridge Building and Software Development." Weblog post. *Herding Cats*. 16 June 2009 <http://herdingcats.typepad.com/my_weblog/2009/06/bridge-building-and-software-development.html>.
25. "The Role of Software Architect and Software Architecture." *Open Frame Technologies*. 8 Oct. 2009 <www.openframetechnologies.com/Vision1.htm>.
26. Spolsky, Joel. "The Perils of Java Schools." *Joel on Software*. 29 Dec. 2005 <www.joelonsoftware.com/articles/ThePerilsofJavaSchools.html>.

Notes

1. Except where noted, all references to CMMI are to CMMI-DEV v1.2, the model. (CMU/SEI-2006-TR-008)
2. See, for example, discussions at "XP and the CMM" <http://c2.com/cgi/wiki?XpAndTheCmm>; Ronald E. Jefferies' "Extreme Programming and the Capability Maturity Model" <www.

xprogramming.com/xpmag/xp_and_cmm.htm>; and "XP and CMM – Agile and Other Processes Forum at JavaRanch" <www.coderanch.com/t/130051/Agile-Other-Processes/XP-CMM>.

3. W. Edwards Deming, known for "Out of the Crisis," and "The New Economics" (widely credited with Total Quality Management); Joseph M. Juran, known for his development of quality planning, quality control, and quality improvement; Philip B. Crosby, known for "Quality Is Free"; to Shigeo Shingo and Taiichi Ohno, for their work with the Toyota Production System, a precursor to Lean manufacturing; and Sakichi Toyoda, known for developing the "5 Whys."
4. My recent informal survey data from several Internet sources where both Agile and CMMI are discussed revealed that most conflicts are perceived to exist from misapplications of Agile and/or CMMI, not from specific content in either.

About the Author

Hillel Glazer has been working in process improvement since 1991. He is one of the few CMMI High Maturity Lead Appraisers and CMMI Instructors working with Agile teams. Glazer serves as an SEI visiting scientist, contributing to the services constellation while also providing Agile-oriented input to CMMI V1.3. He is the lead author on the SEI's first-ever official publication addressing Agile development. His company, Entinex, focuses on bringing Lean and Agile values back into the formalized process improvement to create advanced states of process and performance excellence.

300 E. Lombard ST # 840
Baltimore, MD 21202
Phone: (410) 814-7513
E-mail: hillel@entinex.com